Exercise

Clean up with CTEs

In chapter 2, you generated a list of countries and the number of matches in each country with more than 10 total goals. The query in that exercise utilized a subquery in the `FROM` statement in order to filter the matches *before* counting them in the main query. Below is the query you created:

```
SELECT
  c.name AS country,
  COUNT(sub.id) AS matches
FROM country AS c
INNER JOIN (
  SELECT country_id, id
  FROM match
  WHERE (home_goal + away_goal) >= 10) AS sub
ON c.id = sub.country_id
GROUP BY country;
```

You can list one (or more) subqueries as **common table expressions** (CTEs) by *declaring* them ahead of your main query, which is an excellent tool for organizing information and placing it in a logical order.

In this exercise, let's rewrite a similar query using a CTE.

Instructions 100 XP

- Complete the syntax to declare your CTE.
- Select the `country_id` and match `id` from the `match` table in your CTE.
- Left join the CTE to the league table using `country_id`.

query.sql — Light Mode

```
1  -- Set up your CTE
2  WITH match_list AS (
3      SELECT
4          country_id,
5          id
6      FROM match
7      WHERE (home_goal + away_goal) >= 10)
8  -- Select league and count of matches from the CTE
9  SELECT
10     l.name AS league,
11     COUNT(match_list.id) AS matches
12 FROM league AS l
13 -- Join the CTE to the league table
14 LEFT JOIN match_list ON l.id = match_list.country_id
15 GROUP BY l.name;
```

Run Code | Submit Answer

query result | league | match

league	matches
Switzerland Super League	0
Poland Ekstraklasa	0
Netherlands Eredivisie	1
Scotland Premier League	0

Exercise

Organizing with CTEs

Previously, you modified a query based on a statement you completed in chapter 2 using common table expressions.

This time, let's expand on the exercise by looking at details about matches with very high scores using CTEs. Just like a subquery in `FROM`, you can join tables *inside* a CTE.

Instructions

100 XP

- Declare your CTE, where you create a list of all matches with the league name.
- Select the league, date, home, and away goals from the CTE.
- Filter the main query for matches with 10 or more goals.

Take Hint (-30 XP)

query.sql

Light Mode

```sql
-- Set up your CTE
WITH match_list AS (
  -- Select the league, date, home, and away goals
    SELECT
      l.name AS league,
      m.date,
      m.home_goal,
      m.away_goal,
       (m.home_goal + m.away_goal) AS total_goals
    FROM match AS m
    LEFT JOIN league as l ON m.country_id = l.id)
-- Select the league, date, home, and away goals from the CTE
SELECT league, date, home_goal, away_goal
FROM match_list
-- Filter by total goals
WHERE total_goals >= 10;
```

Run Code | Submit Answer

query result | league | match

league	date	home_goal	away_goal
England Premier League	2011-08-28	8	2
England Premier League	2012-12-29	7	3
England Premier League	2013-05-19	5	5
Germany 1. Bundesliga	2013-03-30	9	2

Exercise

CTEs with nested subqueries

If you find yourself listing multiple subqueries in the `FROM` clause with nested statement, your query will likely become long, complex, and difficult to read.

Since many queries are written with the intention of being saved and re-run in the future, proper organization is key to a seamless workflow. Arranging subqueries as CTEs will save you time, space, and confusion in the long run!

Instructions

100 XP

- Declare a CTE that calculates the total goals from matches in August of the 2013/2014 season.
- Left join the CTE onto the league table using `country_id` from the `match_list` CTE.
- Filter the list on the inner subquery to only select matches in August of the 2013/2014 season.

query.sql Ctrl+O Light Mode

```sql
-- Set up your CTE
WITH match_list AS (
    SELECT
      country_id,
      (home_goal + away_goal) AS goals
    FROM match
    -- Create a list of match IDs to filter data in the CTE
    WHERE id IN (
        SELECT id
        FROM match
        WHERE season = '2013/2014' AND EXTRACT(MONTH FROM date) = 08))


-- Select the league name and average of goals in the CTE
SELECT
    l.name,
    AVG(match_list.goals)
```

Run Code Submit Answer

query result | league | match

name	avg
Switzerland Super League	1.9375000000000000
Poland Ekstraklasa	2.3103448275862069
Netherlands Eredivisie	3.4146341463414634
Scotland Premier League	2.1379310344827586

Exercise

CTEs with nested subqueries

If you find yourself listing multiple subqueries in the `FROM` clause with nested statement, your query will likely become long, complex, and difficult to read.

Since many queries are written with the intention of being saved and re-run in the future, proper organization is key to a seamless workflow. Arranging subqueries as CTEs will save you time, space, and confusion in the long run!

Instructions

100 XP

- Declare a CTE that calculates the total goals from matches in August of the 2013/2014 season.
- Left join the CTE onto the league table using `country_id` from the `match_list` CTE.
- Filter the list on the inner subquery to only select matches in August of the 2013/2014 season.

query.sql

Light Mode

```
12
13
14  -- Select the league name and average of goals in the CTE
15  SELECT
16      l.name,
17       AVG(match_list.goals)
18  FROM league AS l
19  -- Join the CTE onto the league table
20  LEFT JOIN match_list ON l.id = match_list.country_id
21  GROUP BY l.name;
```

Run Code

Submit Answer

query result | league | match

name	avg
Switzerland Super League	1.9375000000000000
Poland Ekstraklasa	2.3103448275862069
Netherlands Eredivisie	3.4146341463414634
Scotland Premier League	2.1379310344827586

Showing 11 out of 11 rows